(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平7－13770

(43)公開日　平成７年(1995)１月17日

(51)Int.Cl.[6]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　9/45				
// Ｇ０６Ｆ　9/34	３３０			

審査請求　未請求　請求項の数２　ＯＬ　（全 12 頁）

(21)出願番号　特願平5－144097

(22)出願日　平成５年(1993)６月16日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(72)発明者　河場　基行
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人　弁理士　井桁　貞一

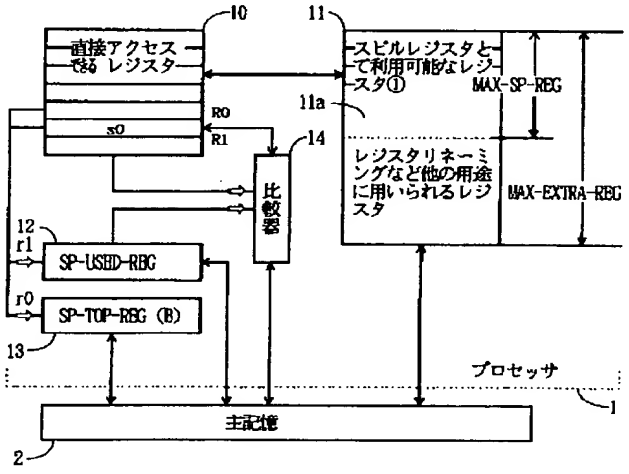
(54)【発明の名称】　ロードストア型プロセッサの付加的レジスタを利用する際のコンパイル方法

(57)【要約】

【目的】　本発明は、プロセッサが備えているマシン命令では指示できない付加的レジスタを有効に利用するためのマシン命令，及び、コンパイル方法に関し、ソースプログラムをコンパイルするときに、プロセッサが備えている上記の付加的レジスタの数に関係なく、有効に利用する。

【構成】　マシン命令では指し示すことができない複数個のスピルレジスタ①を備えたロードストア型プロセッサに、該スピルレジスタ①に配置された変数，又は、主記憶上に割り当てられている変数に対するロード命令②ａと，ストア命令②ｂと，マシン命令で指し示すレジスタに割り当てられなかった変数を、主記憶上に割り当てる際の先頭番地等を所定のレジスタにセットするセット命令②ｃと，スピルレジスタ①他の内容を主記憶 2上の所定の領域に退避する退避命令②ｄと，主記憶 2上の所定の領域に退避されているスピルレジスタ①他の内容を元のレジスタに復帰する復帰命令②ｅとを設けて、コンパイルする。

本発明の原理説明図



1

【特許請求の範囲】

【請求項１】マシン命令では指し示すことができない複数個の付加的レジスタ(11)を備え、上記付加的レジスタ(11)の一部をスピルレジスタ（①）として設定しているロードストア型プロセッサ(1) で実行されるオブジェクトプログラムを生成するコンパイラにおいて、

上記マシン命令で指し示すレジスタ(10)に割り当てられなかった変数を、主記憶(2) 上に割り当て、上記プロセッサ(1) により、上記主記憶(2) 上に割り当てられている変数の一部を上記スピルレジスタ（①）に割り当てられたとき、上記スピルレジスタ（①）に割り当てられている変数と、上記スピルレジスタ（①）に割り当てられないで、主記憶(2) 上に残っている変数に対するロード命令（②ａ）と，ストア命令（②ｂ）と，上記マシン命令で指し示すレジスタ(10)に割り当てられなかった変数を、主記憶(2) 上に割り当てる際の先頭番地(B) と、各プログラムで必要とする上記スピルレジスタ（①）の数を、所定のレジスタ(13, 12) にセットするセット命令（②ｃ）と、上記スピルレジスタ（①）他の内容を主記憶(2) 上の所定の領域に退避する退避命令（②ｄ）と，上記主記憶(2) 上の所定の領域に退避されているスピルレジスタ（①）他の内容を元のレジスタに復帰する復帰命令（②ｅ）を設けると共に、

上記セット命令（②ｃ）に基づいて、上記各プログラムで必要とする上記スピルレジスタ（①）の総数を格納するスピル総数レジスタ(12)と，上記主記憶(2)に割り当てられる変数の先頭番地(B) を格納する先頭番地レジスタ(13)と，上記ロード命令（②ａ），ストア命令（②ｃ）の実行時に、上記主記憶(2) に割り当てられる変数の主記憶(2) 上のレジスタ番号(s0)と、上記各プログラムで必要とするスピルレジスタ（①）の総数を格納するスピル総数レジスタ(12)の内容とを比較して、上記スピルレジスタ（①）に割り当てられている変数と、主記憶(2)上に残っている変数とを、選択的に認識する比較手段(14)とを設けて、

コンパイラが、上記ロードストア型プロセッサ(1) の直接アクセスできるレジスタ(10)に割り当てることが望ましい変数の一部を、上記直接アクセスできるレジスタ(10)に配置できなかったことを検出したとき、上記スピルレジスタ（①）に割り当てられた変数と，上記主記憶(2) 上に割り当てられたままの変数に対するアクセスを、上記セット命令（②ｃ）と、ロード命令（②ａ），ストア命令（②ｂ）とで、選択的に行うようにコンパイルすることを特徴とするロードストア型プロセッサの付加的レジスタを利用する際のコンパイル方法。

【請求項２】上記コンパイル時に、ソースプログラムの主プログラムから、サブルーチンに移った時、該コンパイルされたオブジェクトプログラムの実行時に、上記退避命令（②ｄ）で、上記オブジェクトプログラムの主プログラムで使用していたスピルレジスタ（①）の内容

2

と、上記スピルレジスタ（①）の総数を格納するスピル総数レジスタ(12)と、上記マシン命令が指し示すレジスタ(10)に割り当てられなかった変数を主記憶(2) 上に割り当てる際の先頭番地(B) を格納する先頭番地レジスタ(13)の内容とを、上記主記憶(2) 上の所定の領域に退避するようにコンパイルし、

上記コンパイルされたオブジェクトプログラムの実行時に、上記サブルーチンで必要とする上記スピルレジスタ（①）の数を所定のレジスタ(R1)にセットし、上記スピルレジスタ（①）の内容が主記憶(2) に割り当てられる領域の先頭番地(ベース番地) を所定のレジスタ(R0)にセットした後、上記セット命令（②ｃ）を実行したとき、上記レジスタ(R0)の内容を、上記先頭番地レジスタ(13)にセットした後、当該プロセッサが備えているスピルレジスタ（①）の数が設定されている上記スピル総数レジスタ(12)の内容と、当該サブルーチンで必要とするスピルレジスタの数が設定されているレジスタ(R1) の内容とを、上記比較手段(14)で比較して、当該サブルーチンで必要とするスピルレジスタ（①）の数を確保できるようにコンパイルし、

該オブジェクトプログラム中のサブルーチンの実行が終了した後、上記オブジェクトプログラムのサブルーチン中の復帰命令（②ｅ）で、上記主記憶(2) 上に退避しておいた、上記主ルートプログラムで使用していたスピルレジスタ（①）と、上記スピルレジスタ（①）の総数を格納するスピル総数レジスタ(12)と、上記マシン命令が指し示すレジスタ(10)に割り当てられなかった変数を主記憶(2)上に割り当てる際の先頭番地を格納する先頭番地レジスタ(13)との内容を、それぞれ、元のレジスタ(11, 12, 13)に復帰するようにコンパイルすることを特徴とするロードストア型プロセッサの付加的レジスタを利用する際のコンパイル方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、マシン命令では指し示すことができない複数個の付加的レジスタを備えたロードストア型プロセッサにおいて、上記付加的レジスタの一部をスピルレジスタ①として利用する際のコンパイルに必要なマシン命令，及び、コンパイル方法に関する。

【０００２】通常、プロセッサが実行するマシン命令が主記憶をアクセスする場合、該プロセッサが備えている汎用レジスタへのアクセスに比較して、随分遅い。このことは、プログラムの動作速度を低下させる要因の一つとなっている。そこで、ソースプログラムをコンパイルとき、コンパイラは、できるだけ汎用レジスタを使用する命令コードを出力し、プロセッサのハードウェアは、できるだけ多くのレジスタを利用することが可能なようになっていることが必要となる。

【０００３】又、上記の問題から、最近においては、主記憶に対するアクセスをロード，ストア命令に限定し

て、加算命令等のその他の命令ではレジスタ間演算とする、所謂ロード・ストア型のＲＩＳＣプロセッサとか、スーパースカラプロセッサが構築されている。

【０００４】このようなＲＩＳＣプロセッサとか、スーパースカラプロセッサにおいては、最近のデータ処理の多様化に伴って、レジスタを、より多く利用するプログラムが多くなっている。

【０００５】又、プログラムの各マシン命令間にレジスタ干渉があると、各マシン命令の前後関係を補償してやる必要があるが、例えば、命令１：Ｄ←Ａ＋Ｅ，命令２：Ａ←Ｂ＋Ｃのように、２つの命令１と，命令２との間に順序関係がある場合、後の命令におけるレジスタＡを、例えば、別のレジスタＦにリネーミングしておくことで、上記命令１と命令２との並列処理が可能になり、上記スーパースカラプロセッサ等においては、有用な手段となる。

【０００６】又、同様に、命令１：Ａ←Ｂ＋Ｃ，命令２：Ｇ←Ａ＋Ｆ，命令３：Ａ←Ｄ＋Ｅなる命令列がある場合、命令１と命令２との間には順序関係が必要であるが、命令３のレジスタＡを、例えば、レジスタＨにリネーミングしておくことにより、命令１と命令３との並列処理が可能になる。

【０００７】又、シリアルに構成された命令列（命令１，命令２，～，命令ｋ，～命令ｎ）において、命令Ｋと命令Ｋ＋１との間には順序性を守る必要があるが、命令１，～と、命令Ｋとの間に順序を守る必要がないとき、該命令Ｋを命令１と並列に実行することができる。そうすると、命令Ｋで使用される、例えば、汎用レジスタ(R0)の保持期間が長くなってしまう問題が発生し、この汎用レジスタ(R0)の使用効率が低下する。そこで、該汎用レジスタ(R0)を、例えば、該汎用レジスタ(R0)～とは別に、付加的に設けられているレジスタ群のレジスタＭにリネーミングしておくことで、汎用レジスタ(R0)の使用効率を低下させることなく、且つ、命令１と命令Ｋとの並列処理が可能になる。

【０００８】従って、スーパースカラプロセッサ等においては、レジスタの数が多いと、効率の良い並列処理が可能になる。そこで、前述のように、付加的なレジスタを備えたプロセッサが必要とされることになるが、マシン命令で、上記付加的なレジスタを指定しようとすると、一般には、所定のレジスタ指定フィールドを設けるとか、汎用レジスタ指定フィールドのビット幅を増加させる必要があり、プロセッサの種別に対応して、命令のフォーマットを変える必要が生じ、オブジェクトプログラムの互換性がなくなる等の問題が発生する。

【０００９】このような事情から、プロセッサに付加的なレジスタを備えて、且つ、プロセッサによって、該付加的なレジスタの数が変化しても、コンパイルされたオブジェクトの互換性を保持しながら、且つ、該付加的なレジスタの利用効率を低下させることのないコンパイル

方法が必要とされる。

【００１０】

【従来の技術】図１４は、従来の付加的レジスタに対するアクセス方法を説明する図である。最近のハードウェアの進歩により、プロセッサ内に、通常の命令のレジスタ指定フィールドで指し示すことができる汎用レジスタ(GR) 10 の他に、付加的なレジスタ（前述のスピルレジスタとして使用できるレジスタ群を含む）を備えることができるようになっている。

【００１１】然しながら、プロセッサで実行され、レジスタの多用するオブジェクトプログラムをコンパイラ等で生成する場合、ソースプログラム中の変数で、該汎用レジスタ(GR) 10 に割り当てることが望ましい変数から順次、該汎用レジスタ(GR) 10 に割り当て、該汎用レジスタ(GR) 10 に配置できなかった変数を、主記憶(MS)2 に割り当てることが行われる。

【００１２】

【発明が解決しようとする課題】前述のように、主記憶へのアクセスは、レジスタへのアクセスに比較すると、格段にアクセス速度が遅い。このことは、プログラムの実行速度を低下される要因の一つとなる。

【００１３】一方、前述のように、最近のハードウェア技術の進歩に伴って、より多くのレジスタを、ハードウェア内に用意することができるようになっている。然し、それだけでは、これらのレジスタを利用することはできない。

【００１４】従って、一般には、マシン命令中に、より多くのレジスタを指し示すのに十分なレジスタ指定フィールドが用意されている必要がある。このことは、命令の語長を増やすことと同じである。

【００１５】ハードウェアにインプリメントできるレジスタの数が変化するに伴って、命令語長が変わるのは、マシンコードの互換性を損ねてしまう問題がある。又、前述のように、プログラム中の各マシン命令が使用しているレジスタの間に、出力依存関係，或いは、逆依存関係がある場合、別のレジスタ、即ち、上記ハードウェアで用意されている付加的レジスタにリネーミングすることにより、例えば、スーパースカラプロセッサでの並列処理が容易となる。

【００１６】更に、上記ハードウェアで用意されている付加的レジスタの利用効率を高めるためには、上記コンパイラが主記憶に割り当てた変数を、上記付加的レジスタに再割り当てることができることが必要となるが、このとき、付加的なレジスタの数が変化しても、命令語長が変化しないことが重要となる。

【００１７】然しながら、現状のコンパイラでは、ハードウェア内に上記付加的なレジスタが用意されていても、利用することができないという問題があった。本発明は上記従来の欠点に鑑み、ソースプログラムをコンパイルするときに、プロセッサが備えているマシン命令で

は指示できない付加的レジスタを、マシン命令の語長を変更することなく、効率良く利用する方法を提供することを目的とするものである。

【００１８】

【課題を解決するための手段】図１は、本発明の原理説明図である。上記の問題点は下記の如くに構成されたロードストア型プロセッサの付加的レジスタを利用する際のコンパイル方法によって解決される。

【００１９】(1) マシン命令では指し示すことができない複数個の付加的レジスタ 11 を備え、上記付加的レジスタ 11 の一部をスピルレジスタ①として設定しているロードストア型プロセッサ 1で実行されるオブジェクトプログラムを生成するコンパイラにおいて、上記マシン命令で指し示すレジスタ 10 に割り当てられなかった変数を、主記憶 2上に割り当て、上記プロセッサ 1により、上記主記憶 2上に割り当てられている変数の一部を上記スピルレジスタ①に割り当てられたとき、上記スピルレジスタ①に割り当てられている変数と、上記スピルレジスタ①に割り当てられないで、主記憶 2上に残っている変数に対するロード命令②ａと，ストア命令②ｂと，上記マシン命令で指し示すレジスタ 10 に割り当てられなかった変数を、主記憶 2上に割り当てる際の先頭番地(B) と、各プログラムで必要とする上記スピルレジスタ①の数を、所定のレジスタ 13,12にセットするセット命令②ｃと、上記スピルレジスタ①他の内容を主記憶 2上の所定の領域に退避する退避命令②ｄと，上記主記憶 2上の所定の領域に退避されているスピルレジスタ①他の内容を元のレジスタに復帰する復帰命令②ｅを設けると共に、上記セット命令②ｃに基づいて、上記各プログラムで必要とする上記スピルレジスタ①の総数を格納するスピル総数レジスタ 12 と，上記主記憶 2に割り当てられる変数の先頭番地(B) を格納する先頭番地レジスタ 13 と，上記ロード命令②ａ，ストア命令②ｃの実行時に、上記主記憶 2に割り当てられる変数の主記憶2上のレジスタ番号(s0)と、上記各プログラムで必要とするスピルレジスタ①の総数を格納するレジスタ 12 の内容とを比較して、上記スピルレジスタ①に割り当てられている変数と、主記憶 2に残っている変数とを、選択的に認識する比較手段 14 とを設けて、コンパイラが、上記ロードストア型プロセッサ 1の直接アクセスできるレジスタ 10 に割り当てることが望ましい変数の一部を、上記直接アクセスできるレジスタ 10 に配置できなかったことを検出したとき、上記スピルレジスタ①に割り当てられた変数と，上記主記憶 2上に割り当てられたままの変数に対するアクセスを、上記セット命令②ｃと、ロード命令②ａ，ストア命令②ｂとで、選択的に行うようにコンパイルする。

【００２０】(2) 上記コンパイル時に、ソースプログラムの主プログラムから、サブルーチンに移った時、該コンパイルされたオブジェクトプログラムの実行時に、上記退避命令②ｄで、上記オブジェクトプログラムの主プログラムで使用していたスピルレジスタ①の内容と、上記スピルレジスタ①の総数を格納するスピル総数レジスタ 12 と、上記マシン命令が指し示すレジスタ 10 に割り当てられなかった変数を主記憶 2上に割り当てる際の先頭番地(B) を格納する先頭番地レジスタ 13 の内容とを、上記主記憶 2上の所定の領域に退避するようにコンパイルし、上記コンパイルされたオブジェクトプログラムの実行時に、上記サブルーチンで必要とする上記スピルレジスタ①の数を所定のレジスタ(R1)にセットし、上記スピルレジスタ①の内容が主記憶 2に割り当てられる領域の先頭番地（ベース番地）を所定のレジスタ(R0)にセットした後、上記セット命令②ｃを実行したとき、上記レジスタ(R0)の内容を、上記先頭番地レジスタ 13 にセットした後、当該プロセッサが備えているスピルレジスタ①の数が設定されている上記スピル総数レジスタ 12 の内容と、当該サブルーチンで必要とするスピルレジスタ①の数が設定されている上記レジスタ(R1) 10 の内容とを、上記比較手段 14 で比較して、当該サブルーチンで必要とするスピルレジスタ①の数を確保できるようにコンパイルし、該オブジェクトプログラム中のサブルーチンの実行が終了した後、上記オブジェクトプログラムのサブルーチン中の復帰命令②ｅで、上記主記憶 2上に退避しておいた、上記主ルートプログラムで使用していたスピルレジスタ①と、上記スピルレジスタ①の総数を格納するスピル総数レジスタ 12 と、上記マシン命令が指し示すレジスタ 10 に割り当てられなかった変数を主記憶 2上に割り当てる際の先頭番地を格納する先頭番地レジスタ 13 との内容を、それぞれ、元のレジスタ 11,12,13 に復帰するようにコンパイルするように構成する。

【００２１】

【作用】先ず、本発明においては、コンパイラが、通常は、汎用レジスタ(GR)に割り当てるべきであるにも関わらず、主記憶上に割り当てた変数を、ハードウェアによって、該ハードウェアが備えている付加的レジスタの一部に割り当てられたとき、このレジスタをスピルレジスタ①と呼ぶ。又、上記、ハードウェアが用意している付加的レジスタの数をMAX-EXTRA-REG と定義し、上記スピルレジスタ①の最大数をMAX-SP-REGと定義して、コンパイラによって割り当てるスピルレジスタ①の数は、この値を越えないように制御される。

【００２２】又、本発明においては、ハードウェアとして、次のレジスタを設ける。

SP-TOP-REG：主記憶 2に割り当てられる変数のベース番地（先頭番地で、主プログラム，サブルーチン毎に異なる）B が格納されるレジスタ。

SP-USED-REG ：あるサブルーチン等で必要とするスピルレジスタ①の総数を格納するレジスタで、上記MAX-SP-REGによって制限される。（主プログラム，サブルーチン

毎に異なる）
SP-REG(i)　：付加的なレジスタ 11 の一部であるスピルレジスタ(i) ①で、その総数が上記MAX-SP-REGによって制限される。

【００２３】又、本発明においては、本来プロセッサが備えている命令セットに、以下の５つの命令を設けて、該命令セットを変更する。但し、R0, R1, S0, S1, SP-TOP-REG, SP-USED-REGはレジスタ名であり、r0, r1, s0, s1, sp-top-reg, sp-used-regは、上記各レジスタに格納される値を示すものとする。

【００２４】図２～図９は、本発明のコンパイラで生成されるマシン命令を説明する図であって、図２～図４は、本発明の追加命令の動作を流れ図で示した図であり、図５～図８は、本発明の追加命令に関連するハードウェアの構成図である。

【００２５】以下、各図を参照しながら、追加命令の作用動作を説明する。先ず、図２によって、セット命令②ｃの動作を説明する。
sp-set r0, r1：あるサブルーチンで使用される変数が割り当てられる主記憶 2上の先頭アドレス（ベースアドレス：B)(r0) が、所定のレジスタ(R0)に設定され、当該サブルーチン等で必要とされるスピルレジスタの総数(s0)が、所定のレジスタ(R1)に設定される。

【００２６】本命令 sp-set r0, r1 が、コンパイルされたオブジェクトプログラム上で実行されると、上記先頭アドレス（ベースアドレス:B)(r0) が、上記SP-TOP-REG 13に転送される。又、レジスタ(R1)の値である"r1"が、上記定義されているMAX-SP-REGと比較され、値が小さい方のが、上記SP-USED-REG 12に転送される。つまり、この命令によって、あるサブルーチンをコンパイルするとき、該サブルーチンで必要とするスピルレジスタ①が確保され、前述のリネーミングなどの他の用途に使用されないようにする。〔図２(a) の処理ステップ 100～104 参照〕つまり、レジスタ(R1)の値(r1)が、上記MAX-SP-REGで定義されている付加的レジスタ 11aの数より小さいと、該付加的レジスタ 11aの一部をスピルレジスタ①として確保し、レジスタ(R1)の値(r1)が、上記MAX-SP-REGで定義されている付加的レジスタ 11aの数より大きいときは、上記MAX-SP-REGで定義されている付加的レジスタ 11aの全てをスピルレジスタ①として使用することを意味する。勿論、MAX-EXTRA-REG と MAX-SP-REG が等しければ、付加的レジスタ 11 を、全てスピルレジスタ①として使用することが可能である。

【００２７】次に、図３(a),図５，図６により、ロード命令②ａの動作を説明する。
sp-ld s0, r0　：あるサブルーチンで使用される、ある変数が割り当てられる主記憶 2上の先頭アドレス(B) からの変位 s0 〔上記SP-TOP-REG 13 に設定されている値(r0)＋４s0、但し、主記憶 2の１語が４バイトバウンダリの場合〕が、所定のレジスタ(S0)に設定される。〔図２(b) 参照〕本命令 sp-ld s0, r0が、コンパイルされたオブジェクトプログラム上で実行されると、上記SP USED REG 12の内容(sp-used-reg) 〔つまり、スピルレジスタ①の総数〕と、該サブルーチンで使用される変数の主記憶 2上のアドレス（番地)(s0) とが比較され、0 ≦s0＜SP-USED-REG 12の内容(sp-used-reg) が検出されたときには、スピルレジスタ①に、該変数が割り当てられていることになるので、SP-REG(S0)の内容を、レジスタ(R0)に転送（ロード動作がレジスタ間転送動作となる）する。〔図５，図３(a) の処理ステップ 200, 201参照〕然し、s0≧SP-USED-REG 12の内容(sp-used-reg) が検出されたときには、該変数は、スピルレジスタ①に割り当てられなかったことを意味しているので、主記憶 2上の、上記 sp-top-reg+4s0 番地の値を、レジスタ(R0)にロードする。〔図６，図３(a) の処理ステップ 200, 202参照〕次に、図３(b), 図７，図８によって、ストア命令②ｂの動作を説明する。本命令 sp-st r0, s0が、コンパイルされたオブジェクトプログラム上で実行されると、上記SP USED REG 12の内容(sp-used-reg) 〔つまり、スピルレジスタ①の総数〕と、該サブルーチンで使用される変数の主記憶 2上のアドレス（番地)(s0) とが比較され、0 ＜s0＜SP-USED-REG 12の内容(sp-used-reg) が検出されたときには、スピルレジスタ①に、該変数が割り当てられていることになるので、レジスタ(R0)の内容を、SP-REG(S0)に転送（ストア動作がレジスタ間転送動作となる）する。〔図７，図３(b) の処理ステップ 300, 301参照〕然し、s0≧SP-USED-REG 12の内容(sp-used-reg) が検出されたときには、該変数は、スピルレジスタ①に割り当てられなかったことを意味しているので、レジスタ(R0)の内容を、主記憶 2上の、上記 sp-top-reg+4s0 番地にストアする。〔図８，図３(b) の処理ステップ 300, 302参照〕次に、図４によって、退避命令②ｄ，復帰命令②ｅの動作を説明する。退避命令 sp-save (r0) ：コンパイルされたオブジェクトプログラムの実行途上において、主プログラムからサブルーチンに遷移するとき、該サブルーチンの入口で実行される命令で、前述の sp-set r0, s0 の前に実行されなくてはならない。

【００２８】この命令が実行されると、r0個のスピルレジスタ①が、主記憶 2上の所定の退避領域に退避される。このとき、スピルレジスタ SP-REG(i)①の内容は、主記憶2上のsp-top-reg+i×4(レジスタのバイト数) 番地に退避される。又、SP-TOP-REG 13, SP-USED-REG 12の内容は、例えば、スタックに積むなどの操作によって、主記憶 2上の所定の領域に退避する。〔図４(a) の処理ステップ 400, 401参照〕復帰命令 sp-restore (r0)：コンパイルされたオブジェクトプログラムの実行途上において、主プログラムから遷移したサブルーチンでの処理が終了して、該サブルーチンから主プログラムに復帰するときに実行される命令である。

【００２９】この命令が実行されると、上記 sp-save (r0) によって、主記憶 2上の所定の領域のスタックに積まれていた、SP-TOP-REG 13, SP-USED-REG 12の内容が、それぞれ、元の SP-TOP-REG 13, SP-USED-REG 12 に復帰（ロード）されると共に、主記憶 2上の所定の退避領域に退避されていたスピルレジスタ SP-REG(i)①の内容が、sp-top-reg+i×4(レジスタのバイト数) 番地から、それぞれ、対応するスピルレジスタ①に復帰（リストア）される。〔図４(b) の処理ステップ 500, 501参照〕　本発明のために用意されたセット命令，ロード命令，ストア命令，退避命令，復帰命令は、上記のように動作するので、ソースプログラムをコンパイルするとき、コンパイラは、先ず、従来方法と同じようにして、レジスタ、例えば、通常のプロセッサに設けられている汎用レジスタ(GR) 10 に割り当てることが望ましい変数から順に、該汎用レジスタ(GR) 10 に割り当て、該汎用レジスタ(GR)10 に割り当てることができなかった変数を主記憶の所定の領域に割り当てる。

【００３０】このとき、主プログラム，サブルーチン内において、それぞれ、連続した領域に割り当てるようにする。そのため、主プログラム，各サブルーチンに対応して、前述の先頭アドレス(sp-top-addr) を定義し、上記主記憶上に割り当てる変数は、sp-top-addr+α〔レジスタのバイト数、例えば、前述の４バイトで割り切れる番地で、前述のように、レジスタのバイト数が４バイトであると、４×s0番地〕に割り当てる。

【００３１】そして、コンパイラは、該主記憶上に割り当てられた変数へのアクセスに対して、上記ロード命令(sp-ld s0, r0), 又は、ストア命令(sp-st r0, s0) を生成する。

【００３２】このとき、上記 s0 〔即ち、各サブルーチンで使用される変数の主記憶上のアドレスで、上記sp-top-addr+α で表される〕の値は、コンパイル時に決定される。

【００３３】従って、該コンパイルされたオブジェクトプログラムが、該プロセッサ上で実施され、上記生成されたロード命令(sp-ld s0, r0), 又は、ストア命令(sp-st r0, s0) が実行されると、スピルレジスタ①に割り当てられている変数であるか、否かが、前述の比較手段 14〔図１参照〕で比較され、スピルレジスタ①に割り当てられている変数へのアクセスについては、該スピルレジスタ①との間の転送命令として動作し、該スピルレジスタ①に割り当てられなかった変数に対しては、主記憶上に割り当てられている番地との間で、ロード，ストア動作が実行される。

【００３４】このとき、該スピルレジスタ①へのアクセス命令は、専用命令であり、前述のように、各プロセッサに備えられているスピルレジスタ①の総数(MAX-SP-REG)と、各サブルーチン等で使用される変数〔レジスタ(R1)の内容〕の数によって、動的に、オペランドである s0, r0が生成されるので、ソースプログラムを変更することなく、任意の数のスピルレジスタ①を備えたプロセッサ 1で実行されるオブジェクトプログラムのコンパイルが可能となる。

【００３５】又、退避命令②ｄ，復帰命令②ｅは、主プログラムからサブルーチンに遷移するとき、主プログラムで使用されていたスピルレジスタ①等の内容が破壊される恐れがあるので、主記憶上の所定の領域に退避し、該サブルーチンの実行が終了して元の主プログラムに戻るとき、上記退避しておいたスピルレジスタ①等の内容を元の、対応するレジスタに復帰させるためのものである。

【００３６】上記のように作用するので、本発明によれば、ロードストア型プロセッサが備えている付加的レジスタの使用効率を向上させることができる。特に、使用できるスピルレジスタ①の数を考慮にいれて、マシン命令を生成するプロセッサ、及び、そのプロセッサで実行するオブジェクトプログラムを生成する為のコンパイラの性能の向上に寄与することが大きい。

【００３７】

【実施例】以下本発明の実施例を図面によって詳述する。前述の図１は、本発明の原理説明図であり、図２～図８は、本発明のコンパイラで生成するマシン命令を説明する図であり、図９～１３は、本発明の一実施例を模式的にに示した図であり、図９～図１１は、サブルーチンの実行開始時の動作を示し、図１２，図１３は、サブルーチンの実行終了時の動作を示している。

【００３８】本発明においては、マシン命令では指し示すことができない複数個の付加的レジスタ 11 を備え、上記付加的レジスタ 11 の一部をスピルレジスタ①として設定しているロードストア型プロセッサ 1で実行されるオブジェクトプログラムを生成するコンパイラにおいて、上記マシン命令で指し示すレジスタ 10 に割り当てられなかった変数を、主記憶 2上に割り当て、上記プロセッサ 1により、上記主記憶 2上に割り当てられている変数の一部を上記スピルレジスタ①に割り当てられたとき、上記スピルレジスタ①に割り当てられている変数と、上記スピルレジスタ①に割り当てられないで、主記憶 2上に残っている変数に対するロード命令②ａと，ストア命令②ｂと，上記マシン命令で指し示すレジスタ 10 に割り当てられなかった変数を、主記憶 2上に割り当てる際の先頭番地(B) と、各プログラムで必要とする上記スピルレジスタ①の数を、所定のレジスタ 13, 12にセットするセット命令②ｃと、上記スピルレジスタ①他の内容を主記憶 2上の所定の領域に退避する退避命令②ｄと，上記主記憶 2上の所定の領域に退避されているスピルレジスタ①他の内容を元のレジスタに復帰する復帰命令②ｅを設けると共に、上記セット命令②ｃに基づいて、上記各プログラムで必要とする上記スピルレジスタ①の総数を格納するレジスタ(SP-USED-REG) 12と，上記

主記憶 2に割り当てられる変数の先頭番地(B) を格納するレジスタ(SP-TOP-REG) 13 と，上記ロード命令②ａ，ストア命令②ｃの実行時に、上記主記憶 2に割り当てられる変数の主記憶 2上のレジスタ番号(s0)と、上記各プログラムで必要とするスピルレジスタ①の総数を格納するレジスタ(SP-USED-REG) 12の内容とを比較する比較手段 14 とを設けて、コンパイラが、上記ロードストア型プロセッサ 1の直接アクセスできるレジスタ 10 に割り当てることが望ましい変数の一部を、上記直接アクセスできるレジスタ 10 に配置できなかったことを検出したとき、上記スピルレジスタ①に割り当てられた変数と，上記主記憶 2上に割り当てられたままの変数に対するアクセスを、上記ロード命令②ａ，ストア命令②ｂで、選択的に行うようにコンパイルする手段が、本発明を実施するのに必要な手段である。尚、全図を通して同じ符号は同じ対象物を示している。

【００３９】以下、図１～図８を参照しながら、図９～図１３によって、本発明のロードストア型プロセッサの付加的レジスタを利用する際のコンパイル方法を説明する。先ず、コンパイラは、ソースプログラムの主プログラム，サブルーチンで使用される変数の依存関係を調べ、通常のプロセッサ 1が備えている、例えば、汎用レジスタ(GR) 10 に割り当てることが望ましいにも関わらず、該汎用レジスタ(GR) 10 に割り当てることができなかった変数を、主記憶 2上の所定の領域に割り当てる。但し、これらの変数が、主プログラム，或いは、サブルーチン内では、連続した主記憶 2の領域に割り当てられるようにする。このとき、各々の変数は、レジスタのバイト数（ｍとする）で割り切れる番地に割り当てるが、その領域の先頭番地は、主プログラム，或いは、各サブルーチンで定められている固定されるアドレス(sp-top-addr) となる。

【００４０】つまり、主記憶 2上に割り当てられる変数は、前述のレジスタ(SP-TOP-REG) 13 に、コンパイラによって、予め、設定されているアドレス(sp-top-addr) ＋α（０mod ｍ：レジスタのバイト数ｍで割り切れる番地）に割り当てられる。

【００４１】そして、上記主記憶 2上の｛(sp-top-addr) ＋α｝番地に割り当てられた変数に対して、コンパイラは、スピルレジスタ配置の優先順位をつけて、優先順位の高いものから順に、sp-top-addr+0, sp-top-addr+1,一に割り当て、自己の備えているスピルレジスタ①の総数(MAX-SP-REG)を認識しているハードウェアが、この優先順位に従って配置する。

【００４２】コンパイラは、該スピルレジスタ①に配置された変数に対するアクセスに対して、本発明の上記ロード命令(sp-ld s0,r0), ストア命令(sp-st r0,s0) を生成する。

【００４３】従って、コンパイルされたオブジェクトプログラムを実行し、上記主記憶 2上の｛(sp-top-addr) ＋α｝番地に割り当てられた変数を、レジスタ(R0)にロードするとき、上記本発明のロード命令(sp-ld s0,r0) が実行される。

【００４４】上記ロード命令(sp-ld s0,r0) のオペランドを構成している s0,r0は、前述のようにコンパイル時に決定されているので、s0 (即ち、サブルーチン等で使用されるレジスタ番号) が、前述のセット命令(sp-set r0,r1)により、前述の(MAX-SP-REG)に基づいて設定されているレジスタ(SP-USED-REG) 12の値より小さいときは、該変数が、スピルレジスタ①に配置されていると認識して、該スピルレジスタ｛SP-REG(s0)｝の内容をレジスタ(R0)に転送（ロード）することで、上記ロード命令がレジスタ間転送命令として動作し、高速ロードが行われ、該 s0 がレジスタ(SP-USED-REG) 12より大きいときは、該変数はスピルレジスタ①に配置されていないと認識して、主記憶 2上の｛(sp-top-addr) ＋α｝番地に割り当てられている変数をレジスタ(R0)にロードする。ストア命令(sp-st r0,s0) を実行する場合についても、同様の動作が行われる。｛図５～図８参照｝次に、ソースプログラムの主プログラムからサブルーチンに遷移するときの処理方法を、以下に説明する。

【００４５】主プログラムからサブルーチンに遷移したとき、主プログラムで使用していたスピルレジスタ①の内容が破壊される恐れがある、本発明におけるコンパイル時においては、サブルーチンの入口で、先ず、該スピルレジスタ①と、上記セット命令(sp-set r0,r1)②ｃによって設定されているSP-USED-REG 12, SP-TOP-REG 13の内容とを、本発明の退避命令(sp-save r0)②ｄによって、図９，図１０に示したように、主記憶 2の所定の領域（図１２，図１３の黒の三角印で示す）に退避するようにコンパイルする。

【００４６】その後、前述の sp-top-addr Bを、レジスタ(R0)に設定し、当該サブルーチンで必要とするスピルレジスタの総数をレジスタ(R1)に設定して、本発明の前述のセット命令(sp-set r0,r1)を実行するようにコンパイルする。この結果、コンパイルされたオブジェクトプログラムの実行時において、上記SP-TOP-REG 13, 及び、SP-USED-REG 12に、それぞれ、当該サブルーチンで使用される変数が割り当てられる主記憶 2上の先頭アドレスと、当該サブルーチンで使用されるスピルレジスタ①の総数が設定され、前述のロード命令(sp-ld s0,r0), 或いは、ストア命令(sp-st r0,s0) が実行されるときに参照されることになる。｛図１１参照｝このようにして、該サブルーチンの実行が終了したとき、そのサブルーチンの出口で、上記主記憶 2の所定の領域（図１２，図１３の黒の三角印で示す）に退避されていた、上記SP-USED-REG 12, SP-TOP-REG 13の内容と、主プログラムで使用していたスピルレジスタ①の内容が、前述の本発明で設けられている復帰命令(sp-restore)②ｅが実行されたとき、図１２，図１３に示されているように、それぞれ、

対応するレジスタに復帰するようにコンパイルする。このように、本発明においては、マシン命令では指し示すことができない複数個のスピルレジスタ①を備えたロードストア型プロセッサに、該スピルレジスタ①に配置された変数，又は、主記憶上に割り当てられている変数に対するロード命令②ａと，ストア命令②ｂと，マシン命令で指し示すレジスタに割り当てられなかった変数を、主記憶上に割り当てる際の先頭番地Ｂ等をセットするセット命令②ｃと，スピルレジスタ①他の内容を主記憶 2上の所定の領域に退避する退避命令②ｄと，主記憶 2上の所定の領域に退避されているスピルレジスタ①他の内容を元のレジスタに復帰する復帰命令②ｅとを設けて、コンパイラが、ソースプログラムのコンパイル時に、汎用レジスタに割り当てることが望ましい変数を、汎用レジスタに配置（ハードウェアによる配置）できなかったことを検出したとき、上記スピルレジスタ①に配置されている変数，又は、主記憶上の変数に対するアクセスを、上記各プロセッサが備えているスピルレジスタ①の数に関係なく、ロード命令②ａ，ストア命令②ｂで選択的に行うようにコンパイルする。又、サブルーチンのコンパイル時には、主プログラムで使用されているスピルレジスタ①等が破壊されないように、該スピルレジスタ①等の内容を主記憶上に退避し、該サブルーチンの実行の終了時において、元のレジスタに復帰することができるようにコンパイルするようにした所に特徴がある。

【００４７】

【発明の効果】以上、詳細に説明したように、本発明のロードストア型プロセッサの付加的レジスタを利用する際のコンパイル方法によれば、コンパイラは、レジスタに割り当てることができたにも関わらず、ハードウェア資源の制約のために主記憶に割り当てざるを得なかった変数を認識しているので、それらの変数にスピルレジスタ配置の優先度をつけて、優先度の高いものを、スピルレジスタ①の上位｛sp-top-addr ＋α(4s0) 番地で、優先度の高いものほどαは小さい）番地に割り当てることができ、動作速度の向上を図ることができる。又、スピルレジスタ①に割り当てることができる変数へのアクセスを、本発明の上記 sp-set 命令と、sp-ld 命令と，sp-st命令を用いることによって、ハードウェアの提供した付加的レジスタの数が変化しても、上記sp-set命令が参照する MAX-EXTRA-REGや，MAX-SP-REG に対するインプリメントを変更するだけで、オブジェクトの互換性が保てる。つまり、sp-set命令によって、ハードウェアにあった付加的レジスタの数の変化を吸収することができる。

【００４８】従って、本発明によれば、ロードストア型プロセッサが備えている付加的レジスタの使用効率を向上させることができる。特に、使用できるスピルレジスタ①の数を考慮にいれて、マシン命令を生成するプロセッサ、及び、そのプロセッサで実行するオブジェクトプログラムを生成する為のコンパイラの性能の向上に寄与することが大きい。

【図面の簡単な説明】

【図１】本発明の原理説明図

【図２】本発明のコンパイラで生成するマシン命令を説明する図（その１）

【図３】本発明のコンパイラで生成するマシン命令を説明する図（その２）

【図４】本発明のコンパイラで生成するマシン命令を説明する図（その３）

【図５】本発明のコンパイラで生成するマシン命令を説明する図（その４）

【図６】本発明のコンパイラで生成するマシン命令を説明する図（その５）

【図７】本発明のコンパイラで生成するマシン命令を説明する図（その６）

【図８】本発明のコンパイラで生成するマシン命令を説明する図（その７）

【図９】本発明の一実施例を模式的に示した図（その１）

【図１０】本発明の一実施例を模式的に示した図（その２）

【図１１】本発明の一実施例を模式的に示した図（その３）

【図１２】本発明の一実施例を模式的に示した図（その４）

【図１３】本発明の一実施例を模式的に示した図（その５）

【図１４】従来の付加的レジスタに対するアクセス方法を説明する図

【符号の説明】

1　　プロセッサ

10　　直接アクセスできるレジスタ，汎用レジスタ(GR)

11　　直接アクセスできない付加的レジスタ

12　　必要とするスピルレジスタ①の数を格納するスピル総数レジスタ(SP-USED-REG)

13　　変数を主記憶上に割り当てるときの先頭番地レジスタ(SP-TOP-REG)

14　　比較手段（比較器）

2　　主記憶

100 ～104, 200 ～202, 300 ～302, 400, 401, 500, 501　処理ステップ

①　　スピルレジスタ｛SP-REG(i)｝

②ａ　　ロード命令(sp-ld s0, r0)

②ｂ　　ストア命令(sp-st r0, s0)

②ｃ　　セット命令(sp-set r0, r1)

②ｄ　　退避命令(sp-save r0)

②ｅ　　復帰命令(sp-restore r0)

MAX-SP-REG　　スピルレジスタの総数

MAX-EXTRA-REG　　外部的レジスタの総数

r0　　変数を主記憶上に割り当てるときの先頭番地（ベ

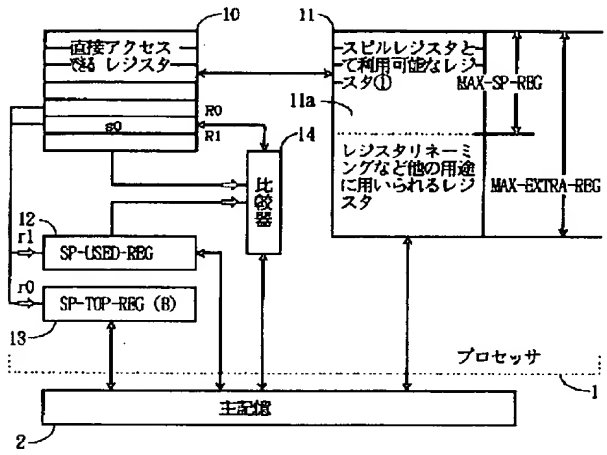
ースアドレス）

r1　必要とするスピルレジスタ①の数

＊s0　主記憶上に割り当てられた変数の番号，変数の変移

【図１】

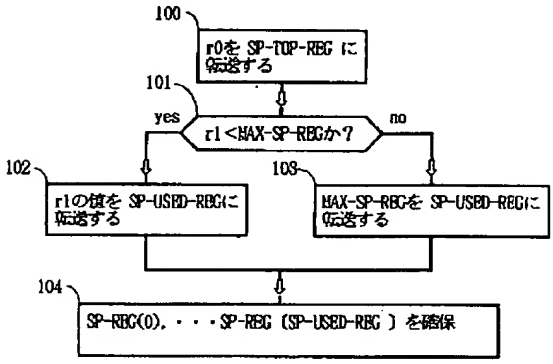
本発明の原理説明図



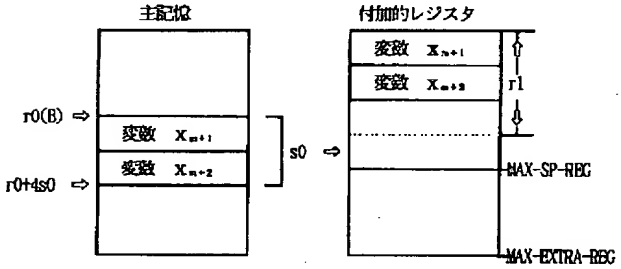
【図２】

本発明のコンパイラで生成する命令を説明する図（その１）

〔sp-set r0,r1〕②c



(a)

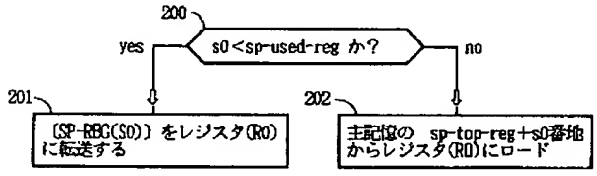


(b)

【図３】

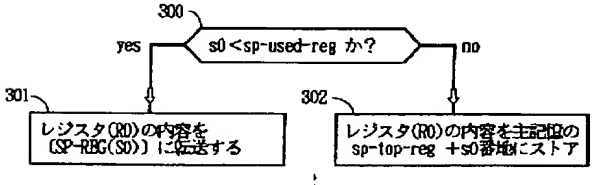
本発明のコンパイラで生成するマシン命令を説明する図（その２）

〔sp-ld s0,r0〕②a



(a)

〔sp-st r0,s0〕②b

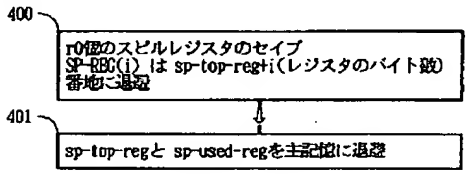


(b)

【図４】

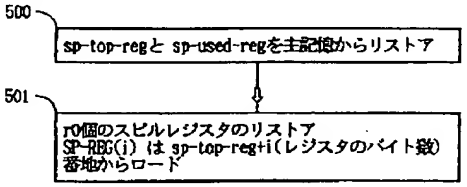
本発明のコンパイラで生成するマシン命令を説明する図（その３）

〔sp-save r0〕②d



(a)

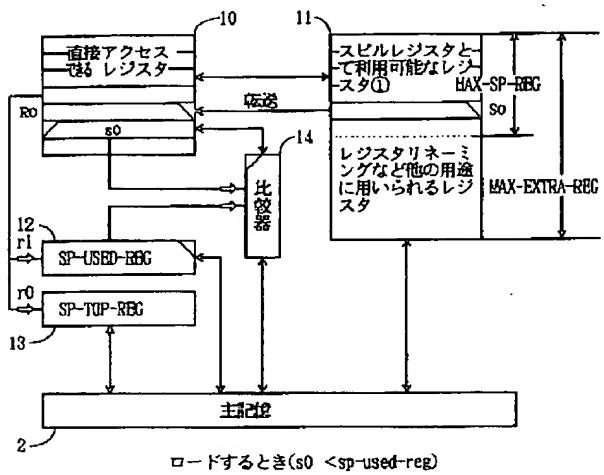
〔sp-restore r0〕②e



(b)

【図５】

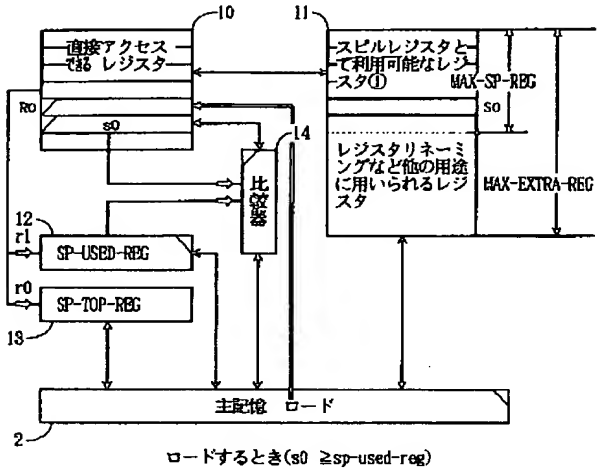
本発明のコンパイラで生成するマシン命令を説明する図（その４）



ロードするとき(s0 <sp-used-reg)

【図６】

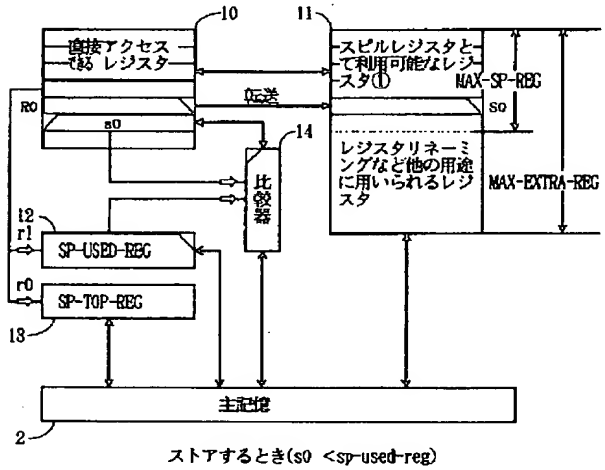
本発明のコンパイラで生成するマシン命令を説明する図（その５）



ロードするとき(s0 ≧sp-used-reg)

【図７】

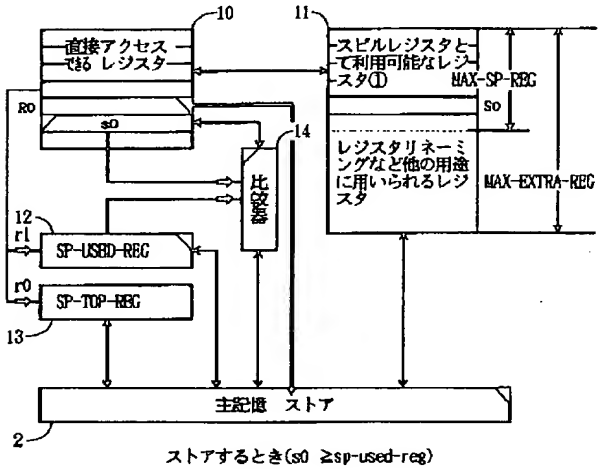
本発明のコンパイラで生成するマシン命令を説明する図（その６）



ストアするとき(s0 <sp-used-reg)

【図８】

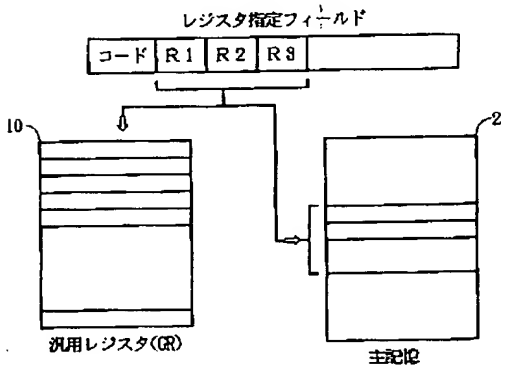
本発明のコンパイラで生成するマシン命令を説明する図（その７）



ストアするとき(s0 ≧sp-used-reg)

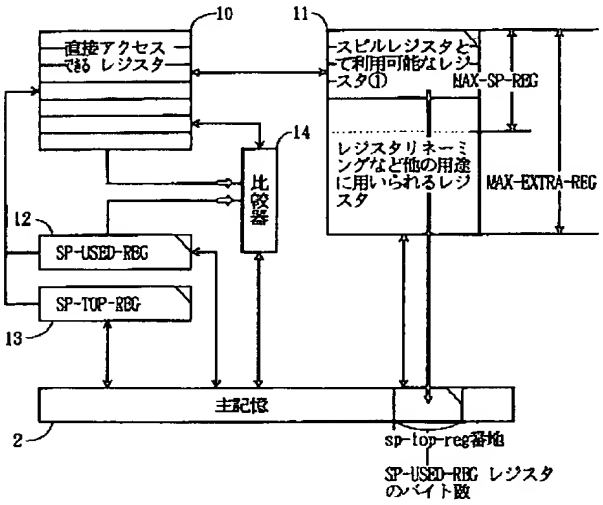
【図１４】

従来の付加的レジスタに対するアクセス方法を説明する図



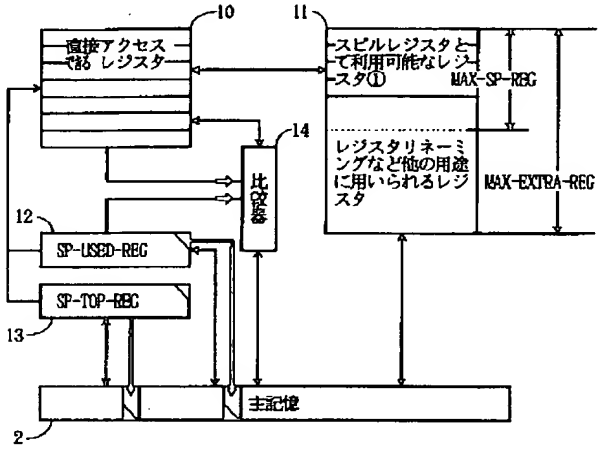
【図９】

本発明の一実施例を模式的に示した図（その１）



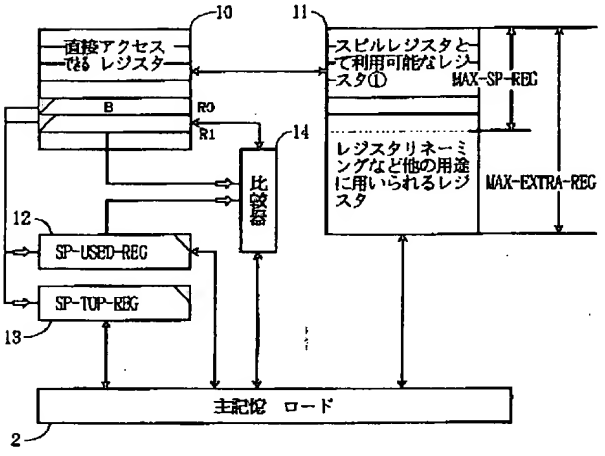
【図１０】

本発明の一実施例を模式的に示した図（その２）



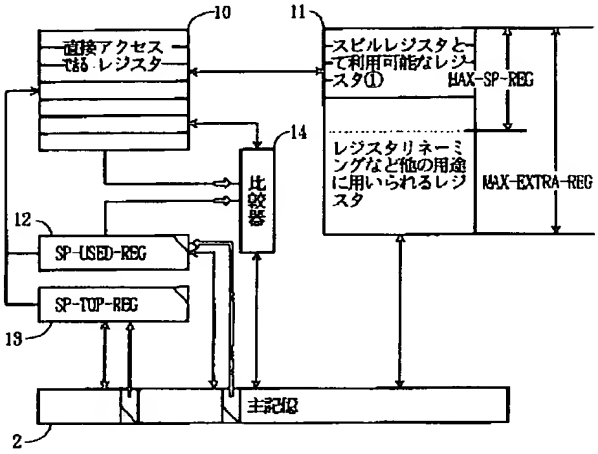
【図１１】

本発明の一実施例を模式的に示した図（その３）



【図１２】

本発明の一実施例を模式的に示した図（その４）



【図１３】

本発明の一実施例を模式的に示した図（その４）